Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Urban Leben

Programska rešitev avtomatizacije procesa sestave turističnega itinerarja

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

prof. dr. Miha Mraz
MENTOR

doc. dr. Miha Moškon
SOMENTOR

Ljubljana, 2017

Tematika naloge:

Kandidat naj v svojem delu zgradi osnutek programske aplikacije za računalniško podprto kreiranje turističnih itinerarjev na osnovi predhodno vnešenih podatkov o posameznih segmentih itinerarja povezanih z izbiro ciljnega mesta obiska. Rešitev naj temelji na minimalistični strojni podpori, primerni za manjše ponudnike turističnih storitev.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom prof. dr. Mihe Mraza in somentorstvom doc. dr. Mihe Moškona,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Urban Leben, Ljubljana, julij 2017.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Urban Leben

Programska rešitev avtomatizacije procesa sestave turističnega itinerarja

POVZETEK

Vse večja dostopnost interneta je prinesla spremembe tudi v turizmu. Povečala se je dostopnost strank do ponudnikovih storitev. Pojavila se je potreba po avtomatizaciji ponavljajočega se ročnega dela in potreba po optimizaciji ter pohitritvi procesov organizacije potovanj na mestu ponudnika. Potrebe večjih podjetij po rešitvah so že potešene, medtem ko manjša podjetja čutijo pomanjkanje primernih orodij.

V pričujočem delu predstavimo osnutek programske rešitve za avtomatizacijo sestave itinerarja. Glavna cilja iskane rešitve sta njena dostopnost in čim boljše razmerje med njeno kompleksnostjo in uporabnostjo. Ciljna skupina naše rešitve so manjša podjetja, ki se ukvarjajo z organizacijo potovanj. Zahteve za končno rešitev izhajajo iz delovnih procesov takih podjetij. S tem želimo premostiti težave ostalih rešitev, ki zahtevajo prilagoditev procesov in omogočiti postopen prehod ter minimalen vpliv na vsakodnevno delovanje ponudnika organizacije turističnih potovanj.

Rešitev predstavljena v diplomskem delu je sistem, ki vključuje centralni strežnik s podatki ter odjemalce, ki poganjajo aplikacijo, ki uporablja te podatke. Strežnik v rešitvi predstavlja Raspberry Pi 3 Model B, odjemalce pa predstavljajo osebni računalniki. Rešitev je implementirana s pomočjo programskega jezika Java.

Ključne besede: sestava turističnega itinerarja, Java, Raspberry Pi

University of Ljubljana
Faculty of Computer and Information Science

Urban Leben

Software solution for the automation of touristic itinerary creation process

ABSTRACT

The incredible growth of the internet has brought changes also to tourism. Clients have gained access to larger pool of providers. Increasing demand for travel services has caused the need for automation and reduction of manual labour in the process of the creation of touristic itineraries. While needs of larger companies have been satisfied, needs of small companies for such software remain unaddressed.

The goal of this thesis is to establish a software solution for the automation of touristic itinerary creation. We aim to find a solution with good complexity versus usability ratio, which would be accessible to smaller companies. Requirements for the solution came from the work processes used in such companies.

The solution we propose addresses the needs of small companies in the process of the creation of touristic itineraries. It presents the system composed of the server with data required for itinerary creation and clients that run an end user application. Server and clients are connected to the same local network. Raspberry Pi 3 Model B is used as the server, and personal computers as its clients. The only requirement imposed on the clients is that they are able to run Java 8 runtime environment, which makes the solution platform independent.

Key words: touristic itinerary creation, Java, Raspberry Pi

ZAHVALA

Zahvaljujem se vsem, ki so me podpirali ob pisanju ter mentorjema za veliko količino potrpežljivosti in pomoči.

— Urban Leben, Ljubljana, julij 2017.

KAZALO

1 Uvod

Globalizacija in internet sta prinesla spremembe tudi v turizmu. Povečal se je bazen potencialnih strank in dostopnost različnih ponudnikov. V odgovor se je razvila potreba po čim večji avtomatizaciji procesov orgranizacije turističnih potovanj in zmanjšanjem ponavljajočega se dela. Velika podjetja so svoje probleme hitro rešila s pomočjo velikih količin sredstev, ki jih imajo na voljo. Probleme so rešila bodisi z lastnimi ali dragimi rešitvami, ki ustrezajo njihovim procesom. Manjša podjetja se še vedno spopadajo s pomanjkanjem primernih rešitev. Rešitve namenjene velikim sistemom so bodisi preveč kompleksne ali pa preprosto predrage, medtem ko so rešitve, ki so na voljo malim podjetjem premalo dovršene.

V majhnih podjetjih pogosto ena oseba opravlja več funkcij in temu primerno potrebuje dostop do več funkcionalnosti naenkrat. Sistemi namenjeni večjim podjetjem težijo k ločitvi posameznih delov procesa, saj ena oseba opravlja samo eno opravilo in potrebuje dostop le do ene funkcionalnosti.

1.1 Cilj rešitve

V pričujočem delu skušamo najti rešitev za avtomatizirano pomoč pri sestavi itinerarja potovanja s čim boljšim razmerjem med kompleksnostjo in uporabnostjo. Rešitev je namenjena manjšim podjetjem, ki se ukvarjajo z organizacijo turističnih potovanj. Končno rešitev skušamo čimbolj približati delovnim procesom takih podjetij. S tem želimo premostiti težave ostalih rešitev, ki zahtevajo prilagoditev delovnih procesov in omogočiti postopen prehod ter minimalen vpliv na vsakodnevno delovanje.

Osredotočamo se na proces izdelave itinerarja in skušamo izdelati rešitev, ki bo ta proces pohitrila ter v čim večji meri avtomatizirala. S tem želimo zmanjšati količino dela in omogočiti večkratno uporabo vnešenih podatkov ter izdelanih itinerarjev. Obenem skušamo čim bolj avtomatizirati tudi izdelavo artefaktov, ki so potrebni za nadaljnjo organizacijo potovanja.

1.2 Pregled dela

V drugem poglavju diplomskega dela razložimo problematiko ter razloge za izdelavo lastne rešitve. Bolj podrobno opišemo proces izdelave itinerarja in možne optimizacije njegove izdelave. Predstavimo obstoječe rešitve in prednosti lastne rešitve.

Tretje poglavje vsebuje predstavitev tehnologij, uporabljenih za izdelavo lastne rešitve in razloge za njihovo uporabo. Opišemo strežnik in storitve, ki jih ponuja. Predstavimo tehnologijo uporabljeno za izdelavo odjemalca.

Četrto poglavje opisuje izdelano rešitev. Predstavljena je implementacija na strani strežnika in odjemalca. Opis strežnika vključuje ponujene storitve in strukturo podatkov v podatkovni bazi. Na strani odjemalca je opisano delovanje odjemalca in proces sestave itinerarja s pomočjo aplikacije.

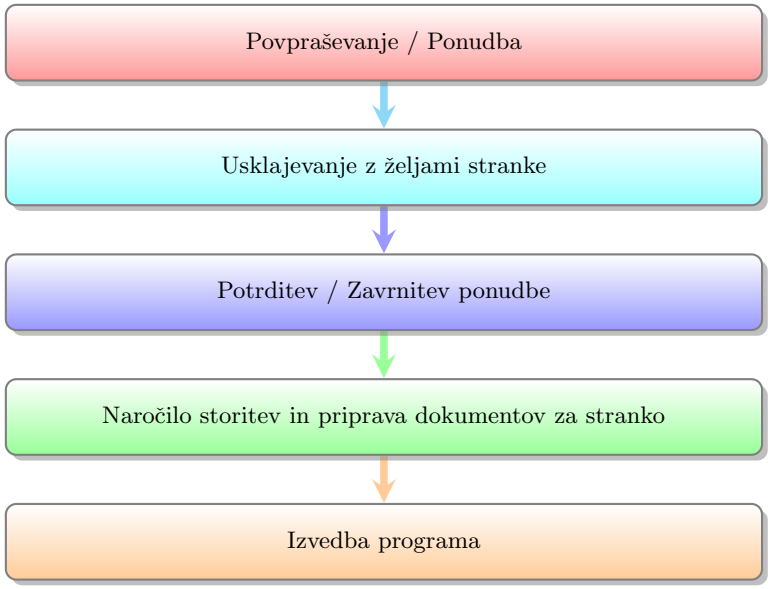
2 Opis problema

Internet in globalizacija sta tudi v turizmu prinesla veliko sprememb. Na eni strani se je povečala dostopnost strank in trgov, na drugi strani pa se je povečala konkurenca. Direktna posledica tega je, da je število povpraševanj in pripravljenih ponudb naraslo, vendar pa se je razmerje povpraševanj proti številu končnih strank povečalo, saj je danes enostavno poslati povpraševanje več kot eni agenciji.

2.1 Proces organizacije potovanja

Proces organizacije potovanja za stranko se začne s tem, ko potencialna stranka pošlje na agencijo povpraševanje, agencija pa nato kot odgovor pošlje ustrezno ponudbo. V primeru, da ima potencialna stranka dodatne želje, se ponudbo nato še prilagaja. Potencialna stranka se nato odloči za ponudbo, lahko pa jo tudi zavrne. V primeru potrditve ponudbe se nato naroči storitve in pripravi dokumente za stranko. Ob ustreznem času pride tudi do izvedbe samega programa. Diagram poteka postopka dela je prikazan na sliki 2.1.

Ponudba, ki je odgovor na povpraševanje, je za agencijo najbolj dolgotrajen in drag



Slika 2.1 Delovni process od ponudbe do izvedbe potovanja.

del organizacije potovanja. Temu botruje dejstvo, da je potrebno stranko še pridobiti in ji aranžma prodati, saj se šele po ponudbi odloči za potovanje. Posledično je število ponudb vedno večje od števila prodanih potovanj, zato je nujno, da ta korak vzame čimmanj časa. Optimizacija tega se ne sme zgoditi na račun kvalitete in upoštevanja želja potencialne stranke, saj je od tega odvisna prodaja ter z njo zaslužek. Prvotna ponudba poslana stranki se nato tudi spreminja in prilagaja željam potencialne stranke, kar pomeni še več vloženega časa in dela v nekaj, kar ne zagotavlja dodatnih prihodkov.

S tem ko se organizacija potovanja naroči, potencialna stranka postane stranka. Agencija nato naroči storitve pri svojih partnerjih ali poišče ponudnika za storitve, ki jih potrebuje. Stranki pripravi voucher[1] oz. dokument o vnaprejšnjem plačilu storitev. Največkrat se za vsakega ponudnika storitev za stranko pripravi en voucher z naštetimi storitvami. Skupaj z voucherji stranka dobi še dokončni itinerar[2]. Na koncu stranka odide na potovanje.

[1] **voucher** - dokument o vnaprejšnjem plačilu določenih storitev, ki ga izda turistična agencija uporabniku

[2] **itinerár** - tur. podroben načrt, popis nameravanega potovanja

2.2 Možnosti poenostavitve procesa s programsko opremo

Poenostavitev procesa je mogoča že v prvem koraku, kjer se agencije srečujejo z velikim številom izdelanih ponudb, ki so osnutki itinerarjev. Ponudbe so oblikovno enake, vsebina pa se spreminja glede na lokacijo, ki jo želi stranka obiskati. Pri tem je v okviru mesta ali lokacije nabor ponudbe omejen in predvidljiv, zato je smiselna in možna poenostavitev izdelave itinerarja za ponudbo. Čas izdelave ponudbe se lahko tako s pomočjo predlogov in predhodno pripravljene vsebine drastično skrajša.

Itinerar skozi nadaljnje usklajevanje z željami stranke obdrži enako obliko, zato bi bila količina dela že z optimizacijo prvega koraka manjša, saj se ob popravkih vnese le vsebinske spremembe, ali pa se uporabi druge pripravljene vsebine. Pri tem za oblikovanje dokumenta poskrbi sistem.

Z uvedbo ustrezne podatkovne baze se lahko poenostavi tudi naslednje korake. Vanjo se lahko shrani servise in pripadajoče stroške. Tako je izpis servisov in stroškov trenuten, saj lahko sistem po sestavljenem itinerarju zbere vse potrebne postavke. V bazo se lahko poleg servisov shrani tudi potrebne dodatne podatke o podizvajalcih, tako da jih sistem navede že ob izpisu. S tem se lahko pohitri naročilo storitev in pripravo dokumentov tako za partnerje kot za stranko. Tudi vse nadaljnje dokumente lahko pripravi sistem in jih po potrebi celo pošilja naprej.

Tak sistem lahko torej pohitri vse korake od povpraševanja do izvedbe. Od rešitve pa je odvisno, ali je poenostavitev celostna, ali le delna in kakšen vpliv ima na obstoječ delovni proces.

2.2.1 Doprinos v hitrosti

Največji doprinos namenske programske opreme je seveda zmanjšanje količine dela potrebnega za pripravo in izvedbo potovanja. Čas potreben za izvedbo nekega koraka se lahko nekajkrat skrajša, kar močno poveča produktivnost posameznega zaposlenega. To se zrcali tako v nižjih stroških za podjetje kot tudi v konkurenčnosti podjetja, saj so odzivnejši za stranke.

2.2.2 Vzdrževanje konsistence med dokumenti

Podjetju ni treba skrbeti, da bodo dokumenti enaki in da bodo ustrezali vsem pravilom in oblikovnim napotkom, saj so dokumenti narejeni z namensko programsko opremo vedno enaki ne glede na število avtorjev.

2.2.3 Preprečevanje človeških napak

Pri tako obsežnih delovnih procesih hitro pride do človeške napake, bodisi nekdo nekaj pozabi všteti, česa ne prepiše ali pozabi pripraviti kak dokument. Faktor človeške napake je nemogoče odpraviti, lahko pa se njegove vplive minimizira. Prav pri tem nam pomaga namenska programska oprema, saj so pogosto za ljudi zahtevne operacije za računalnik enostavne in se pri njih ne moti.

2.3 Pregled obstoječih rešitev

V pričujočem razdelku so opisane obstoječe rešitve, ki olajšujejo delo turističnim agencijam. Opisane so njihove prednosti in slabosti ter razlogi za implementacijo lastne rešitve.

2.3.1 iTravel software

iTravel software je kombinacija sistema za rezervacije in organizacijo potovanj za turistične agencije. Ponuja široko paleto funkcionalnosti, kot so pomoč pri najemu servisov, generiranje določenih dokumentov, iskanje servisov po njihovi bazi in še druge [1]. iTravel software je aplikacija, ki rešuje težave velikih podjetij in ima zato primerno ceno, ki se začne pri nekaj tisoč evrih na leto in raste glede na velikost sistema.

2.3.2 Travefy

Travefy je Ameriško podjetje, ki ponuja spletno aplikacijo za izdelavo itinerarja v obliki spletne strani ali aplikacije za mobilne naprave. Njena glavna prednost je skupinsko urejanje potovanja, ki je primerno za lastno organizacijo potovanja. Pomaga beležiti in deliti stroške [2]. Slabost tega sistema je osredotočenost zgolj na lepoto izdelanega itinerarja in ne ponuja funkcionalnosti za druge dele procesa.

2.4 Prednosti lastne rešitve

Na eni strani imamo polno funkcionalen sistem kot je iTravel Software, ki določa procese za nas, na drugi strani pa sistem kot je Travefy, ki optimizira samo en korak v delovnem procesu. Zato ima pogosto lastna rešitev določene prednosti.

2.4.1 Prilagodljivost

Ena od prednosti lastne rešitve se kaže v tem, da jo lahko prilagodimo svojemu delovnemu procesu. Spremembe dela v prehodu na nov proces so zgolj minimalne in se pojavijo v obliki menjave aplikacije, v kateri delamo. Uporaba sistema kot je iTravel software pa s seboj prinese še prilagoditev delovnega procesa, saj drugače funkcionalnosti, ki nam jih sistem ponuja, ostanejo neizkoriščene.

2.4.2 Konsistenca z obstoječimi dokumenti

V primeru lastne rešitve lahko vse dokumente prilagodimo svojim željam in obdržimo obstoječo obliko. Slednje predstavlja poenostavitev tako za podjetje kot za partnerje, ki te dokumente uporabljajo. To je pomembno predvsem pri dokumentih, kot je npr. voucher, ki se uporabljajo namesto plačilnega sredstva.

2.4.3 Postopnost prehoda

Lastna rešitev nam zaradi zgoraj naštetih lastnosti, kot sta prilagodljivost in konsistenca, omogoča postopen prehod na nov sistem. Delovni proces lahko na nov sistem prestavljamo korak za korakom. To olajša prehod tudi na ta način, da ni zaradi morebitnih težav z novim sistemom ali neskladnosti ogrožen celoten proces, ampak zgolj en korak.

2.4.4 Dostopnost

Aplikaciji kot sta Travefy in iTravel software sta v obliki spletne strani. To pomeni, da je hitrost delovanja omejena s hitrostjo internetne povezave in kvaliteto le-te. Za delovanje aplikacije je potrebna internetna povezava. Omejitvena dejavnika pri delu z njima sta čas nalaganja ter odzivnost. Medtem ko aplikacija, ki teče na lokalnem strežniku ali pa iz njega pobira podatke potrebne za delovanje teče bistveno hitreje kot aplikacije, ki delujejo prek interneta. Časovnega zamika v tem primeru praktično ni. Še vedno je za oddaljeno delo potreben internet, vendar je delo v pisarni, kjer se strežnik nahaja, bistveno hitrejše.

2.4.5 Podatki

Lasten sistem lahko napolnimo s podatki po izbiri, bodisi iz obstoječih baz, razpredelnic, ali pa jih vnesemo na roke. To omogoča dober nadzor nad podatki, ki jih naša aplikacija uporablja. Lahko jih prilagodimo svoji uporabi ali pa jih omejimo na zanesljive ponudnike

oz. partnerje s katerimi delamo. Poleg tega lastna baza omogoča boljši nadzor nad svojimi podatki z vidika zasebnosti. Znano je, da velika podjetja uporabljajo podatke svojih strank za optimizacijo svojih storitev in druge raziskave, pogosto proti njihovi volji, ali pa jim to vsilijo kot pogoje uporabe. V primeru lastne rešitve teh težav ni, saj je lažje nadzirati kdo ima dostop do podatkov in kako so varovani.

2.5 Izdelava lastne rešitve

Pri izdelavi rešitve smo se osredotočili na proces sestave itinerarja, ker predstavlja največ ponavljajočega dela in so mogoči največji prihranki pri količini dela. Takšna rešitev nam ob primerni implementaciji prinaša zgoraj naštete prednosti. Vendar na račun zgornjih optimizacij ne smejo trpeti drugi delovni procesi v turistični agenciji. V ta namen smo pri izbiri tehnologij za izdelavo lastne rešitve skušali zadostiti sledečim pogojem:

- **Optimalno koriščenje obstoječih virov**, kot so računalniki in programska oprema.
- **Podprtost na različnih sistemih**, kot so Windows, Linux in MacOS.
- **Cenovna dostopnost** velja predvsem za programsko opremo, kjer se želimo zanašati na odprtokodne rešitve, saj imajo licence za komercialne rešitve pogosto astronomske cene.
- **Hramba podatkov** v obliki podatkovne baze.
- **Verjetnost uporabe** končne rešitve v praksi. Izdelava rešitve, ki je nihče ne bo uporabljal, je potrata časa in denarja.
- **Verjetnost implementacije** pomeni izbira primernih tehnologij, ki bodo omogočale enostavno implementacijo in dolgoročno podporo rešitvi.
- **Vzdrževanje sistema** tekom njegove življenjske dobe, ki se odraža v posodabljanju ter podpori izbrane programske opreme in zagotavljanju delovanja strojne opreme.

Ti pogoji nam, poleg enostavne integracije v obstoječe delovne procese turistične agencije pomagajo zagotoviti enostavno vsakodnevno uporabo rešitve.

3 Izbira tehnologij

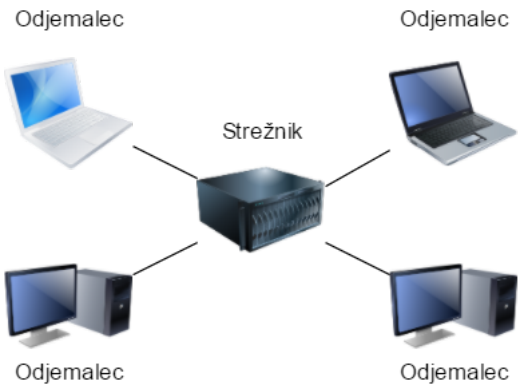
V pričujočem poglavju so opisane tehnologije izbrane za izdelavo programske rešitve prestavljene v 2. poglavju. Pri izbiri tehnologij smo želeli zadostiti glavnim pogojem vzpostavljenim v razdelku 2.5.

3.1 Arhitektura odjemalec strežnik

Za komunikacijo po omrežju bomo uporabili arhitekturo odjemalec-strežnik, ki predpostavlja, da je vsaka naprava v omrežju bodisi strežnik ali odjemalec. Najpogosteje je v enem omrežju več odjemalcev in en ali več strežnikov. Primer tovrstne arhitekture je predstavljen na sliki 3.1. Prednosti te arhitekture so povezljivost, skalabilnost, prilagodljivost, podatkovna integriteta, dosegljivost, učinkovitost ter redundantnost [3].

3.1.1 Strežnik

Strežnik je en ali več računalnikov v omrežju, ki običajno omogočajo različne storitve ter servise odjemalcem v omrežju, kot sta deljenje datotek in povezovanje na podatkovno bazo. V večjih sistemih se za potrebe strežnika uporablja zmogljivejše računalnike, ki



Slika 3.1 Primer arhitekture odjemalec strežnik.

podatke obdelujejo in pošiljajo hitreje od običajnih računalnikov. V manjših sistemih lahko vlogo strežnika prevzamejo tudi manjše naprave kot je Raspberry Pi, ki ga opišemo v razdelku 3.2.

3.1.2 Odjemalec

Odjemalec v omrežju je računalnik, na katerem se izvaja uporabniški vmesnik, sprejema odzive uporabnika, zagotavlja predstavitev podatkov ter povezovanje s storitvami podatkovne baze [3]. To nalogo danes izvaja že večina naprav povezanih v omrežje. V našem primeru so to predvsem računalniki, ki poganjajo operacijske sisteme Windows, Linux ter MacOS.

Za potrebe razvoja in testiranja te naloge je bil uporabljen prenosni računalnik z naloženim operacijskih sistemom Windows 10. V omrežju predstavlja vlogo odjemalca in ima sledeče specifikacije:

- Intel i7 4720HQ procesor,
- 12 GB RAM-a,
- grafična kartica GeForce 840M,
- 1 gigabit-na mrežna kartica.

3.2 Raspberry Pi

Raspberry Pi je serija računalnikov na eni plošči razvitih v Veliki Britaniji za namen širše dostopnosti učenja programiranja v šolah in državah v razvoju [4]. Sistem je doživel veliko

boljše sprejetje za uporabo na področju robotike in domače rabe. Uporabljajo se lahko kot medijski ali spletni strežniki, za hrambo podatkov in še v mnoge druge namene. Do leta 2017 je bilo prodanih že več kot 10 milijonov Raspberry Pi-jev [5].

3.2.1 Računalnik na eni plošči

Izraz računalnik na eni plošči (angl. *single board computer*) označuje tiskano vezje s procesorsko, spominsko in vhodno-izhodno enoto, kar mu omogoča, da deluje kot računalnik. Glavna razlika takega sistema od navadnega računalnika je, da se ti sistemi ne zanašajo na razširitve za periferne funkcije, ampak imajo komponente integrirane ter omogočajo manj možnosti za razširitve. Sprva so se taki sistemi uporabljali za demonstracijo, razvoj in učenje ter kot vgrajeni sistemi.

Današnji računalniki na eni plošči so postali mogoči z manjšanjem velikosti tiskanih vezij. Sistem na enem vezju zmanjša ceno celotnega sistema z zmanjšanjem števila potrebnih tiskanih vezij, števila priključkov in vodil, ki bi bile potrebne za večji sistem. Z združevanjem funkcij na eno vezje se doseže tudi zmanjšanje velikosti celotnega sistema. Poleg tega se izboljša zanesljivost sistema, saj so priključki pogost vir težav pri zanesljivosti [6].

3.2.2 Raspberry Pi 3 Model B

Najnovejši model v družini Raspberry Pi je Raspberry Pi Zero W. Izšel je februarja 2017. Po velikosti je enak svojim predhodnikom, od njih pa se razlikuje v tem, da ima vgrajen brezžični modul, bluetooth modul ter močnejši procesor [7]. Raspberry Pi 3 Model B je prikazan na sliki 3.2 in ima sledeče značilnosti:

- 64-bitni štirijederni procesor iz družine ARMv8 s frekvenco 1.2GHz,
- 1GB RAM-a,
- grafično jedro,
- 801.11n brezžični modul,
- Bluetooth 4.1 in BLE[1] modul,
- 4 USB vhode,

[1] BLE - Bluetooth Low Energy

Slika 3.2 Raspberry Pi 3 Model B.

- HDMI izhod itd.

Tak sistem ob splošni uporabi predstavlja podobno procesno moč, kot jo je imel procesor Intel 300MHz Pentium 2 z več spomina in boljšo grafiko. Za delovanje porabi okoli 10W energije, ter se ne pregreva. Osnovni spomin sistema, na katerem je shranjen operacijski sistem, predstavlja spominska kartica, nadaljnje razširitve pa so mogoče bodisi z USB ključki ali diski. Z velikostjo škatlice kart predstavlja uporabno rešitev kot kompakten strežnik za enostavne naloge in manjše obremenitve [8].

Raspberry Pi bo v našem sistemu prevzel nalogo strežnika, skrbel za podatkovno bazo in deljenje datotek. Arhitektura odjemalec strežnik nam omogoča, da v primeru potrebe po močnejšem strežniku Raspberry Pi zamenjamo z močnejšim računalnikom brez sprememb na ostalih sistemih.

3.2.3 Raspbian

Za Raspberry Pi obstaja uradni operacijski sistem imenovan Raspbian, ki bazira na distribuciji Linux Debiana. Optimiziran je za delovanje na ARM procesorjih, ki jih uporabljajo računalniki Raspberry Pi. S prilagoditvijo izboljša delovanje operacijskega

sistema na takšnih napravah, največja pa je pohitritev operacij s plavajočo vejico [9]. Raspbian je kot večina Linux sistemov odprtokoden in ne potrebuje plačljive licence za uporabo.

3.2.4 Samba

Deljenje datotek med Linux in Windows operacijskima sistemoma prek mreže ne deluje samo po sebi, zato je za deljenje datotek potrebna dodatna programska oprema. Samba je brezplačna programska oprema, ki reši ta problem in omogoči deljenje datotek med Linux in Windows sistemi prek omrežja.

3.3 Relacijska baza

Za hrambo podatkov, ki se uporabljajo v aplikaciji, je najprimernejša relacijska baza. Relacijska baza je podatkovna zbirka, ki deluje na podlagi modela relacij. Podatki so razporejeni v tabele, vsaka tabela pa ima vrstice, stolpce in relacije. Relacije v tabeli predstavljajo logične povezave z drugimi tabelami. Vrstice v tabelah predstavljajo entitete, stolpci pa njihove atribute. Vsaka entiteta ima unikatni ključ, ki se uporablja za enolično določanje entitete in relacije z drugimi entitetami.

Takšna podatkovna baza je primerna za podatke, pri katerih se entitete ne spreminjajo in so jasno strukturirane. Konsistenca podatkov je vzdrževana in zahtevana na strani baze, sprememba lastnosti ali oblike entitete pa tako zahteva spremembo baze. Na drugi strani jasna struktura entitet omogoča hitrejše iskanje po entitetah in relacijah med njimi [10].

3.3.1 Strukturirani povpraševalni jezik

Interakcija z relacijsko bazo se izvaja s pomočjo SQL-a ali strukturiranega povpraševalnega jezika. To je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami. Leta 1986 je postal del standarda ANSI Inštituta, njegov razvoj pa poteka še danes. Osnovni elementi jezika so sledeči [11]:

- stavki, ki so osnovni gradniki poizvedb in izrazov,
- izrazi, ki vrnejo ali skalarne vrednosti ali tabele,
- predikati, ki opisujejo pogoje, ki jih zna SQL ovrednotiti,

- poizvedbe, ki vrnejo podatke glede na izbrane kriterije,
- ukazi, ki izvajajo operacije na bazi itd.

3.3.2 MySQL

MySQL je odprto kodni sistem za nadzor relacijskih baz. Prvotno ga je razvilo švedsko podjetje MySQL AB, sedaj pa je v lasti Oracle Corporation. Zaradi razširjene uporabe, podprtosti ter odprtokodnosti je primeren za uporabo v naši aplikaciji. Izvorna koda je na voljo pod GNU GPL[2] licenco. Obstajajo tudi plačljive verzije z dodatnimi funkcionalnosti, vendar zaradi dimenzij projekta teh funkcionalnosti ne potrebujemo in njihov strošek ne bi bil smiseln [12]. Na Raspberry Pi se naloži v obliki servisa, ki nato prek mreže omogoča dostop do relacijske baze.

3.4 Java

Java je višjenivojski, splošnonamenski, sočasen, objektno orientiran programski jezik, ki ga je leta 1995 razvilo podjetje Sun Microsystems, kasneje pa ga je kupilo podjetje Oracle Corporation. Trenutno je najbolj uporabljan jezik na svetu. Temu pripomore tudi dejstvo, da se v njem pišejo programi za operacijski sistem Android [13].

Večji del sintakse v Javi izhaja iz programskih jezikov C in C++ z nekoliko drugačno organizacijo ter vplivom drugih programskih jezikov [14]. Glavni gradnik Jave je razred. Vsi razredi izhajajo iz osnovnega razreda Object in dedujejo njegove funkcije ter lastnosti.

Java je teče povsod, kjer je mogoče naložiti javansko izvajalno okolje imenovano JRE[3], ki ga je danes mogoče namestiti na skoraj vseh sodobnih sistemih. Trenutno je najnovejša verzija Java 8, ki je v jezik prinesla nekatere lastnosti funkcijskih jezikov.

3.4.1 Sistemske zahteve Java 8

Za poganjanje izvajalnega okolja Java 8 je potrebno imeti računalnik z naloženim operacijskim sistemom Windows, Linux ali MacOS. Obenem mora imeti strojno opremo, ki mora imeti vsaj sledeče performančne lastnostni:[15]:

- 128 MB RAM-a,
- 124 MB prostega prostora na trdem disku,

[2] GNU GPL - GNU General Public License
[3] JRE - angl. Java Runtime Environment

- procesor boljši od Pentium 2 266 MHz.

Zgoraj naštete zahteve so veliko manjše od tega, kar ima danes povprečen računalnik kar pomeni, da je kodo, ki je napisana v Javi mogoče poganjati na praktično vseh računalnikih, ki so danes v uporabi.

3.4.2 Prenosljivost

Glavna ideja pri nastanku Jave je bila prenosljivost in idejni pristop "Write once, run everywhere", ki teži k temu, da je potrebno kodo zgolj enkrat prevesti in jo je možno poganjati kjerkoli, neodvisno od strojne opreme in operacijskega sistema.

To so razvijalci Jave dosegli v obliki javanskega izvajalnega okolja. V okviru JRE teče javanski virtualni stroj ali JVM[4]. Izvorna koda javanskega programa se prevede v strojno kodo imenovano Java Bytecode, katero javanski virtualni stroj interpretira in zanjo priskrbi izvajalno okolje. JVM deluje tudi kot most med Javo in sistemom ter tako skrbi za nizkonivojske funkcije in sistemske klice.

3.4.3 Hitrost

Javansko izvajalno okolje je bilo poleg največje prednosti dolgo časa tudi ena glavnih težav Jave, saj virtualni stroj, ki interpretira kodo predstavlja za računalnik dodatno delo. Posledica tega je bila, da so javanski programi skoraj vedno porabili več spomina in tekli počasneje, kot tisti napisani v podobnih jezikih, kot sta C in C++. To so rešili z različnimi optimizacijami predvsem pa z uporabo sprotnega prevajanja ali JIT prevajanja[5]. JIT prevajanje kodo, ki je pogosto uporabljena in kritična, optimizira tako, da je danes Java po hitrosti izvajanja primerljiva z ostalimi programskimi jeziki.

3.4.4 Nadzor nad spominom

Ena od glavnih nalog javanskega izvajalnega okolja je nadzor nad pomnilnikom. Uporabnik objekte instancira in uporablja, okolje pa poskrbi, da objekte, ki niso več v uporabi, odstrani iz pomnilnika. To preprečuje pojav puščanja pomnilnika (angl. *memory leak*). Puščanje pomnilnika je pojav, ko objekti, ki niso več v uporabi, ostajajo v pomnilniku in povzročajo vedno večjo porabo le tega.

[4] JVM - Java Virtual Machine
[5] JIT - Just In Time

3.4.5 Standardne knjižnice

Javansko izvajalno okolje poleg virtualnega stroja poskrbi še za standardne knjižnice, ki poskrbijo za osnovne operacije kot so grafika, nitenje, zbirke in omrežne operacije. Glavni paketi knjižnic so sledeči [16]:

- **java.awt** - vključuje razrede za delo z grafičnim vmesnikom in slikami,
- **java.io** - poskrbi za vhodno-izhodne operacije z sistemom,
- **java.lang** - vsebuje razrede, ki so temeljni za javo kot jezik,
- **java.math** - vsebuje implementacije matematičnih funkcij,
- **java.net** - poskrbi za omrežne operacije,
- **java.util** - priskrbi implementacije za zbirke in vse podporne funkcije,
- **java.swing** - poskrbi za preprost javanski uporabniški vmesnik, ki je v večjem delu prenosljiv itd.

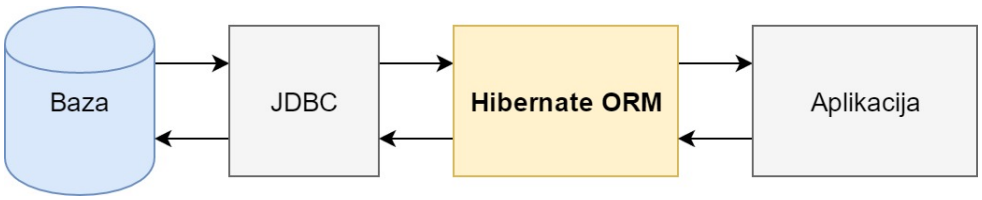
Poleg zgoraj naštetih Java vključuje še mnogo drugih standardnih knjižnic, ki služijo različnim namenom. Vsem tem knjižnicam je kot tudi vsem drugim knjižnicam napisanim v Javi enako, da so prenosljive in tečejo na vseh sistemih, ki podpirajo Javo.

3.4.6 Objektno-relacijske preslikave

Javi omogoča povezovanje na podatkovno bazo gonilnik, ki upošteva JDBC[6] vmesnik. Ta definira, kako se odjemalec povezuje na podatkovno bazo. V prvi vrsti je JDBC vmesnik namenjen povezavam na relacijske baze. Gonilnik nam najpogosteje priskrbi proizvajalec, kar velja tudi v primeru MySQL baze.

Vendar pa tudi taka uporaba baze v Javi zahteva veliko dela, saj se podatki v SQL bazi nahajajo v obliki tabel. Iskanje po njih zahteva poizvedbe, ki so lahko zelo dolge in zahtevne, sploh če vključujejo relacije čez več tabel. Poleg tega podatke iz baze dobimo v obliki tabel, ki jih je potrebno nato še obdelati ter pretvoriti v smiselne strukture znotraj programskega jezika. Za poenostavitev tega dela se uporablja objektno relacijska preslikava. To je programerska metoda za pretvorbo podatkov med nekompatibilnimi podatkovnimi sistemi v objektno orientiranih programskih jezikih.

[6] JDBC - Java Database Connectivity



Slika 3.3 Diagram poteka komunikacije od baze do aplikacije pri uporabi Hibernate ORM-ja.

V Javi za MySQL to nalogo opravi Hibernate ORM, ki je na voljo pod GNU GPL licenco. Hibernate ORM je orodje za objektno relacijsko preslikave v obliki javanske knjižnice. Njegovi glavni nalogi sta preslikava javanskih razredov v tabele na relacijski bazi in s tem preslikava podatkov ter obstojnost podatkov. Skrbi tudi za klice in iskanje po relacijski bazi, pri tem pa avtomatsko generira potrebne ukaze in poizvedbe. Po potrebi omogoča tudi napredne funkcionalnosti kot so transakcije, leno nalaganje (angl. *lazy loading*), obenem pa ne zahteva dodatnih polj ali konfiguracij na relacijski bazi. Konfigurira se ga s pomočjo XML datotek in je tako ločen od razredov in ne zahteva njihovih sprememb [17]. Skrbi za celotno komunikacijo z bazo in funkcionalnosti razvijalcu ponudi v obliki javanskih funkcij in razredov kot je to predstavljeno na sliki 3.3.

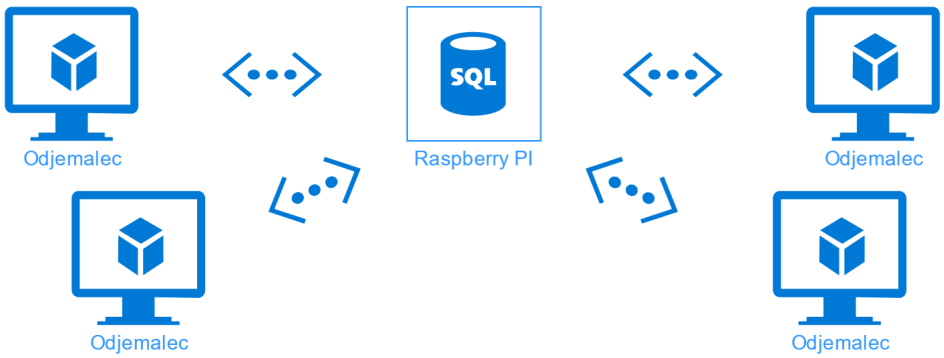
4 Praktična implementacija rešitve

V pričujočem poglavju je predstavljena implementacija avtomatizacije procesa izdelave itinerarjev turistične agencije. Sistem, ki je osnova rešitve problema, vključuje strežnik in več odjemalcev. Strežnik v našem sistemu predstavlja Raspberry Pi, odjemalce pa vse naprave, predvsem osebni računalniki, ki lahko poganjajo Javo 8. Odjemalci poganjajo aplikacijo, ki se povezuje na relacijsko bazo. Slednja teče na strežniku. Shema sistema je predstavljena na sliki 4.1.

4.1 Strežnik

V našem sistemu je vlogo strežnika prevzel Raspberry Pi Model 3. Na njem teče MySQL relacijska baza, verzija 5.5 in servis Samba, ki skrbi za deljenje datotek preko omrežja. Zaradi varnosti Raspberry Pi omogoča povezave zgolj z lokalnega omrežja, tako na storitve relacijske baze, kot tudi na storitve deljenja datotek. To pomeni, da je aplikacijo mogoče uporabljati le, ko smo povezani v lokalno omrežje. V primeru, ko uporabnik želi do aplikacije dostopati oddaljeno, potrebuje VPN[1], ki mu omogoča oddaljeno povezova-

[1] VPN - Virtual Private Network



Slika 4.1 Shema sistema, ki predstavlja rešitev.

nje na lokalno omrežje s pomočjo namenske programske opreme.

4.1.1 Samba

Servis Samba skrbi za deljenje dveh map. Prva je namenjena splošnemu deljenju datotek, druga pa je namenjena datotekam, ki so potrebne za nemoteno delovanje aplikacije, npr. datoteke potrebne za prikazovanje slik v aplikaciji.

4.1.2 Shema podatkovne baze

Podatkovna baza, ki teče na MySQL instanci na Raspberry Pi-ju, predstavlja osnovo za delovanje aplikacije. V njej so podatki, ki služijo izdelovanju različnih dokumentov. Relacijska shema podatkovne baze je prikazana na sliki 4.2.

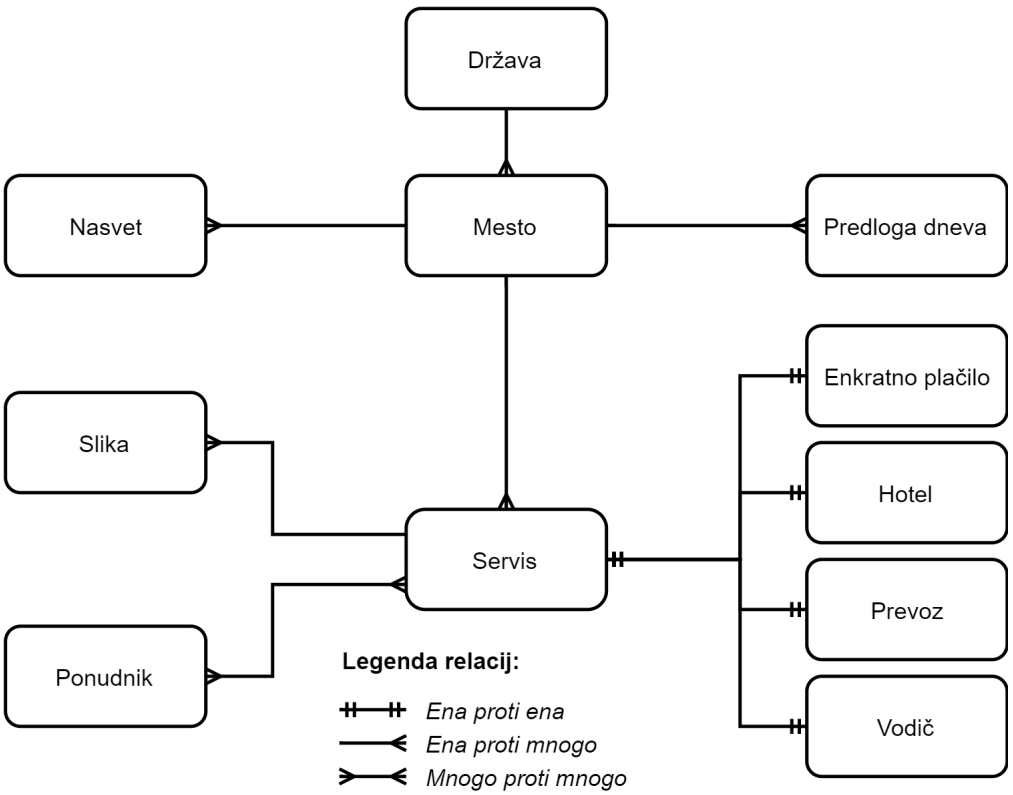
Glavna entiteta v naši podatkovni bazi je **država**. Predstavlja glavno kategorijo, pod katero so podatki naprej razdeljeni v **mesta**. Ideja za to strukturo leži v naravi potovanja, saj stranka ponavadi pride v neko **mesto** in nato tam uporablja različne **servise**. **Servisi** na voljo v posameznem **mestu** so jasni in jih je lahko definirati. S kategorizacijo po **mestih** poenostavimo kasnejše iskanje.

Država

Država je preprosta entiteta v podatkovni bazi. Sestavljata jo zgolj ime in primarni ključ. Služi kot glavni ključ za kategorizacijo **mest**.

Mesto

Mesto je podobno kot **država** preprosta entiteta z imenom in primarnim ključem, vendar je veliko bolj pomembno, saj se pod njega kategorizira večina ostalih entitet. Pod **mesta**



Slika 4.2 Relacijska shema podatkovne baze.

se kategorizirajo **servisi**, **nasveti** in **predloge dni**. Kot **mesto** se lahko uporablja tudi lokacija z večjim številom **servisov**, ki jih je smiselno shraniti v ločeno kategorijo.

Servis

Servis predstavlja osnovno plačljivo postavko, ki se deli na 4 glavne podskupine. To so **hotel**, **prevoz**, **vodič** in **enkratno plačilo**. Vsebuje podatke, ki so skupni vsem naštetim entitetam, kot so lokacija ali mesto, cena, kontaktni podatki itd. Entiteta **servis** je povezana tudi s pomožnima entitetama **slika** in **ponudnik**. Vsak **servis** se kategorizira v eno od glavnih podskupin s povezavo na entiteto, ki ima podatke specifične za vsako podskupino. Podskupine pomagajo predvsem pri filtriranju različnih vrst **servisov** ter pri računanju stroškov, saj se različne vrste servisov različno zaračunavajo. V nadaljevanju so naštete in opisane glavne podskupine **servisov**:

- **Hotel** predstavlja ponudnika nočitve. Vsebuje relevantne podatke kot npr. vrsta sobe, število zvezdic, daljši opis, povezavo do spletne strani ipd.

- **Prevoz** je entiteta, ki hrani podatke specifične za storitev prevoza.
- **Vodič** predstavlja storitev vodenja v mestu ali na lokaciji.
- **Enkratno plačilo** je univerzalen servis. To so lahko obroki v restavracijah, vstopnine in vsi ostali servisi, ki so plačljivi in ne spadajo v ostale kategorije.

Ponudnik

Ponudnik v podatkovni bazi predstavlja možne ponudnike za posamezen **servis**. Teh je lahko več, obenem pa lahko en ponudnik izvaja več servisov. Slednje se uporablja v knjigovodske namene, da se posameznemu servisu lahko pripiše izvajalca. Ta podatkovna zbirka vsebuje podatke o podjetjih.

Slika

Entiteta **slika** predstavlja fotografijo, ki je relevantna za nek **servis**. Vsak **servis** ima lahko več **slik**, ki se nato lahko uporabljajo pri izdelavi itinerarja. Fotografije niso shranjene v podatkovni bazi, temveč kot datoteke na Raspberry Pi-ju in sicer v posebni mapi, ki jo deli Samba. **Slika** ima shranjeno v podatkovni bazi zgolj lokacijo fotografije na Raspberry Pi-ju, do katere se nato dostopa kot do datoteke. Relacijska baza namreč ni namenjena shranjevanju binarnih objektov kot so fotografije, zato jih shranjujemo ločeno.

Nasvet

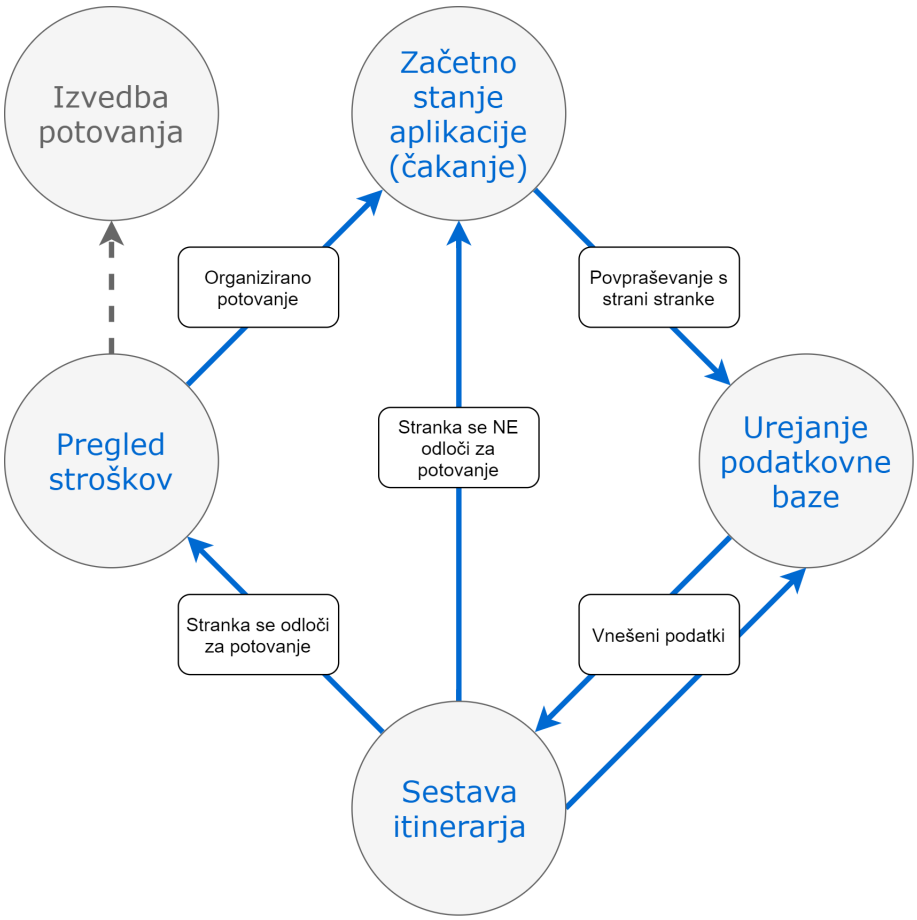
Nasvet predstavlja neplačljivo postavko, ki vsebuje opis neke značilnosti ali predlogo dejavnosti v **mestu**.

Predloga dneva

Predloga dneva omogoča shranjevanje celotnega dneva vnešenega v sistem in s tem poenostavi ponovno uporabo vnešenega dneva.

4.2 Odjemalec

Odjemalec predstavlja aplikacijo napisano v Javi, ki teče na osebnem računalniku in se preko omrežja povezuje na Raspberry Pi. Naenkrat je lahko na Raspberry Pi povezanih več odjemalcev. Uporabniški vmesnik aplikacije je zgrajen s pomočjo standardne javanske



Slika 4.3 Diagram prehodov stanj med koraki uporabe aplikacije.

knjižnice Swing. Aplikacija ima tri glavne funkcionalnosti. Prva je urejanje podatkov v podatkovni bazi, druga pa sestava itinerarja, ki na koncu omogoča izvoz itinerarja v PDF dokument. Tretja funkcionalnost je pregled okvirnih stroškov ter njihov izvoz v format Microsoft Excel.

4.2.1 Aplikacija

Aplikacije omogoča tri glavne funkcionalnosti, ki predstavljajo korake v procesu izdelave itinerarja. Diagram prehodov med njimi je predstavljen na sliki 4.3. Vsak od korakov omogoča več dejanj in pogledov. Shema pogledov in dejanj v okviru vsakega koraka je predstavljena na sliki 4.4. Uporaba pogledov in dejanj je bolj podrobno opisana v razdelkih, ki sledijo.

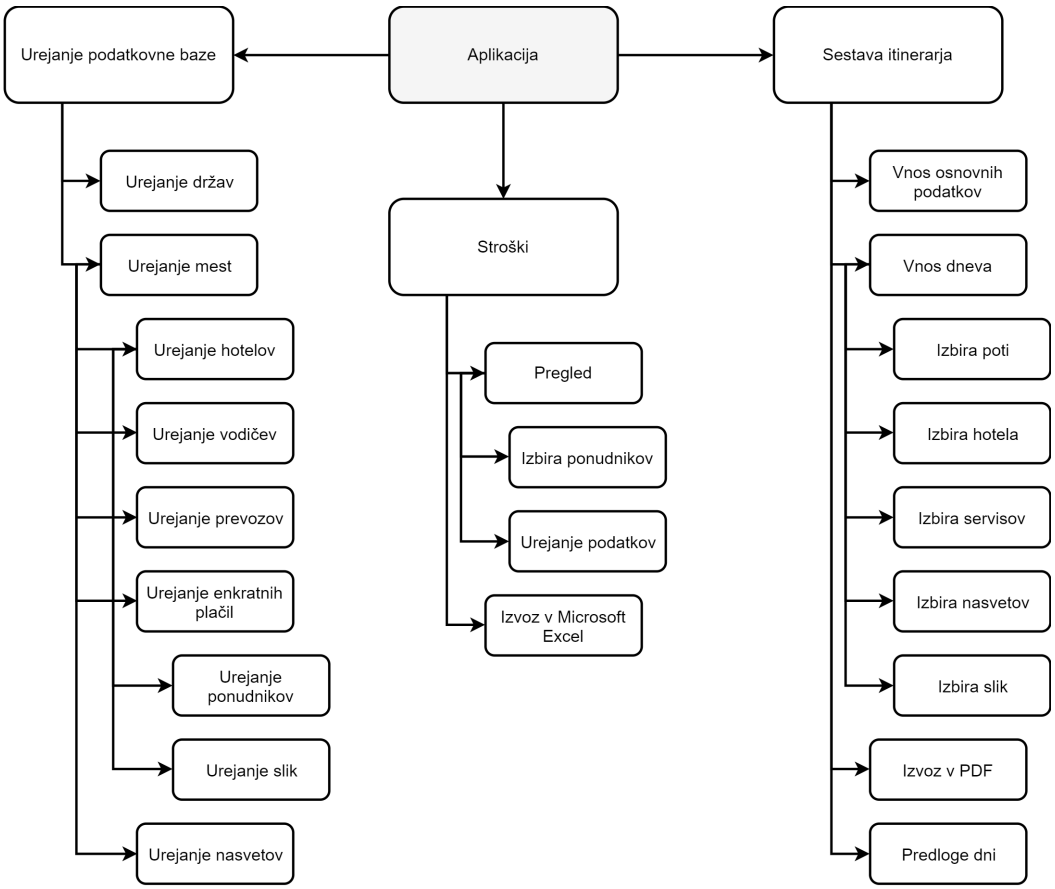
Proces uporabe aplikacije se začne s *čakanjem* na povpraševanje s strani stranke. Po prejetem povpraševanju sledi korak *urejanja podatkovne baze*. V tem koraku se po potrebi podatkovno bazo dopolni z manjkajočimi podatki. V primeru, ko so vsi podatki že vnešeni v podatkovno bazo, se lahko ta korak preskoči. Vnešeni podatki so pogoj za začetek *sestave itinerarja*. Korak vključuje izdelavo itinerarja s pomočjo podatkov v podatkovni bazi in izvoz itinerarja v PDF format. V primeru manjkajočih podatkov je mogoča vrnitev na korak *urejanja podatkovne baze*. Po koncu *sestave itinerarja* se stranka bodisi odloči za potovanje, ali pa ne. V slednjem primeru se proces vrne na korak *čakanja*. Po odločitvi stranke za potovanje se nadaljuje s korakom *pregled stroškov*, rezultat katerega je razpredelnica s postavkami v Microsoft Excel formatu. Od tu naprej se organizacijo preda procesom, ki niso zajeti v sklopu te aplikacije, proces uporabe aplikacije pa se vrne na korak *čakanja*.

4.2.2 Urejanje podatkovne baze

Za uspešno sestavo itinerarja je potreben obstoj podatkov v podatkovni bazi. Sprva vnos podatkov predstavlja dodatno delo, ki pa potem, ko je podatkovna baza napolnjena s pogosto uporabljanimi podatki, predstavlja veliko pohitritev. Nadzor podatkov nam omogoča pogled za urejanje podatkovne baze.

Pogled za urejanje podatkovne baze omogoča dodajanje in brisanje držav, mest ter servisov. S tem nam omogoča gradnjo strukture v obliki držav in pripadajočih mest, ki predstavljajo kategorije, v katere se dodajajo servisi. Pod vsakim mestom so servisi ločeni na 4 glavne kategorije in nasvete. Uporabniški vmesnik z opisano strukturo je predstavljen na sliki 4.5.

Vsaka od glavnih kategorij ima lasten dialog za dodajanje in urejanje pripadajočih servisov. Dialog za eno od glavnih skupin je predstavljen na sliki 4.6. Ta dialog nam omogoča vnos vrednosti, ki so potrebne za posamezno glavno kategorijo. Prisoten je tudi gumb, ki odpre dialog za izbiro ponudnika ter gumb za dialog, ki omogoča dodajanje in odstranjevanje relevantnih slik. Slike, ki jih dodamo, se pretvorijo v zmanjšano obliko in shranijo na Raspberry Pi, da lahko v kasnejših korakih do njih dostopajo vsi uporabniki aplikacije.



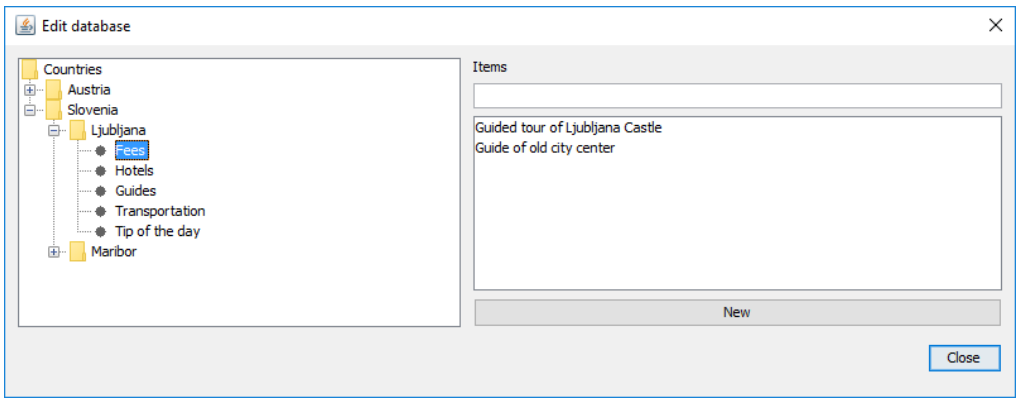
Slika 4.4 Diagram pogledov in funkcij pod vsakim delom aplikacije.

Hibernate konfiguracija

Skrb za komunikacijo s podatkovno bazo je prepuščena knjižnici Hibernate. Konfiguracija zanjo je shranjena v XML označevalnem jeziku. Vsak objekt, ki ga Hibernate pridobi iz baze, ima še lastno konfiguracijsko datoteko, ki povezuje polja na javanskem objektu z stolpci iz podatkovne baze. Poti do teh datotek navedemo na koncu datoteke z osnovno konfiguracijo. Hibernate vzpostavi povezavo s podatkovno bazo ob zagonu aplikacije in jo vzdržuje, dokler se aplikacija ne zapre.

4.2.3 Sestava itinerarja

Glavni korak, katerega aplikacija poenostavi, je sestava itinerarja. Rezultat tega koraka sta itinerar ter seznam stroškov potovanja z oceno končne cene.



Slika 4.5 Pogled za urejanje podatkovne baze. Drevesna struktura je vidna na levi, seznam servisov v kategoriji pa na desni.

Osnovni podatki

Korak sestave itinerarja se začne z vnosom osnovnih podatkov o potovanju. To so podatki o datumu potovanja in stranki ter ciljni državi. Ti podatki so pomembni, saj jih program uporablja pri generaciji izhodnih dokumentov. Pogled za vnos osnovnih podatkov v aplikaciji je predstavljen na sliki 4.7.

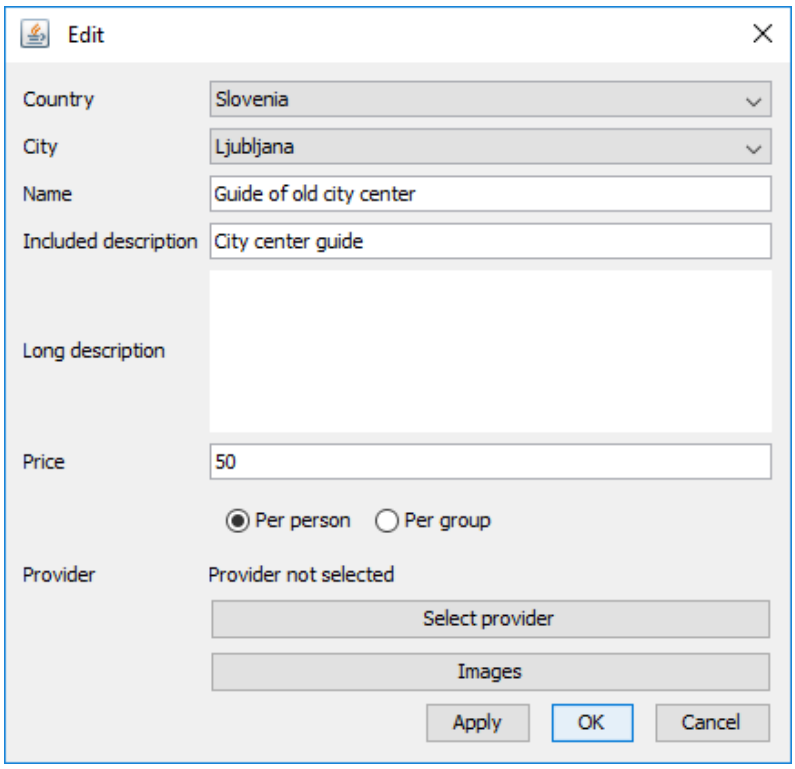
Vnos dneva

Glavni del koraka je vnos podatkov za posamezen dan. Potrebno ga je ponoviti za vsak dan potovanja in zato predstavlja največ dela. Pogled v aplikaciji je predstavljen na sliki 4.8. Ključni del pri vnosu dneva je izbira poti, ki jo opravimo z dialogom, ki nam omogoča izbiro mest iz podatkovne baze. Izbrana mesta nam nato predstavljajo filter za izbiro servisov in nasvetov.

Izbiro prenočišča nam odpre dialog, kjer lahko izbiramo prenočišča iz baze. V tem dialogu so na voljo le prenočišča v mestih na poti, kar olajša iskanje prenočišč. Poleg tega nam omogoča tudi izbiro obrokov, ki so ponujeni v hotelu.

Jedro itinerarja je predstavljeno v obliki tabele. Nahaja se na desnem delu slike 4.8, ki nam omogoča vnos opisa programa. Uporabnik lahko vnese posamezne opise in pripadajoče čase. Funkcionalnost je poenostavljena tako, da nam posamezna vrstica omogoča izbiro servisov, pri katerih se nato v itinerarju uporabi opis shranjen v podatkovni bazi in nam s tem prihrani delo. Servisi so ponovno omejeni z mesti izbranimi kot pot. To nam olajša iskanje ter predstavlja ideje, kaj lahko stranke na izbrani destinaciji počnejo.

Vsi vnosi in izbire v tem pogledu vplivajo na končno obliko itinerarja in izpisane

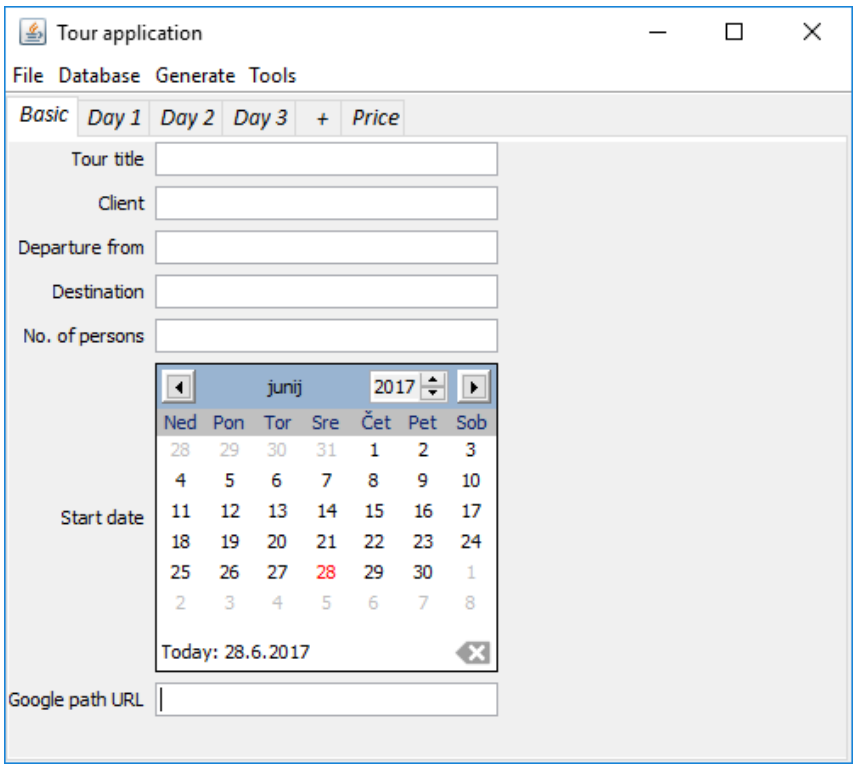


Slika 4.6 Pogled za urejanje servisa.

podatke. Estetska olepšava je omogočena z dialogom za dodajanje slik. Na izbiro so slike, ki so povezane z izbranimi servisi. Na voljo je tudi vnos neobveznih podatkov, kot sta naslov dneva, nasvetov in dodatne informacije za izbrani dan.

Uporaba bi v praksi izgledala tako, da uporabnik vnese naslov dneva in izbere mesta, ki jih bo stranka na izbrani dan obiskala. Sledi izbira hotela ter servisov. Nato zaključi z izbiro relevantnih slik. Te korake ponovi za vsak dan potovanja.

Ta korak ob mnogih ponovitvah predstavlja veliko količino dela, zato program omogoča shranjevanje sestavljenih dni. V praksi to pomeni, da lahko uporabnik zaključen dan shrani kot predlogo. Ta predloga ima lastno ime ter mesto pod katero je kategorizirana. Ob naslednji uporabi lahko uporabnik v novem potovanju uporabi za dan to predlogo iz katere se skopirajo vsi podatki. Tako se lahko izdelava celotnega itinerarja skrajša na vnos osnovnih podatkov in izbiro predlog.

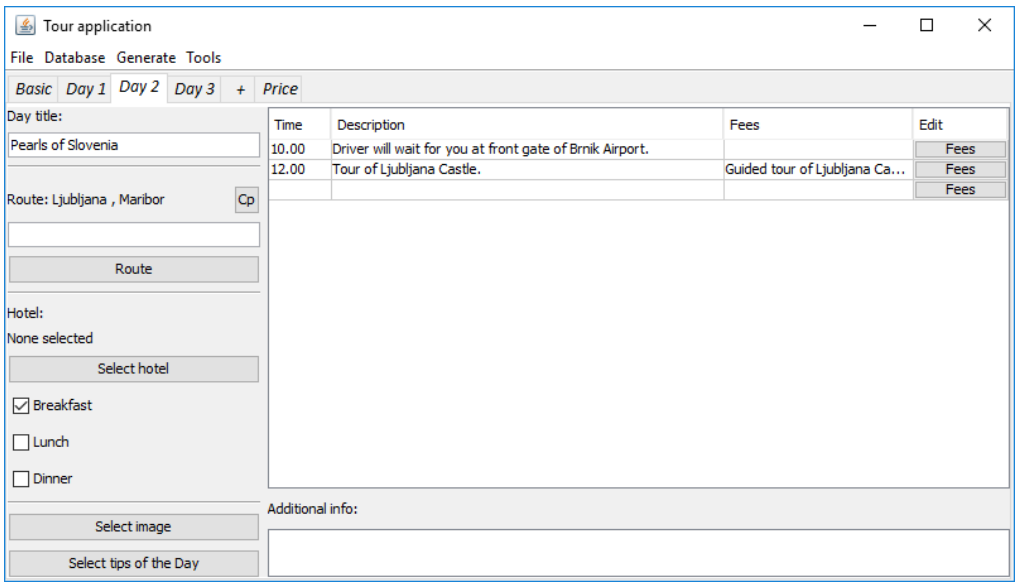


Slika 4.7 Pogled za vnos osnovnih podatkov.

4.2.4 Itinerar

Rezultat zgoraj opisanega koraka je itinerar, ki se izvozi v PDF obliko. Struktura je predhodno določena tako, da vsi itinerarji izvoženi iz sistema izgledajo zelo podobno in imajo konsistentno obliko. Na prvi strani so izpisani osnovni podatki o potovanju. Na drugi strani se generirajo dnevi po predlogi, ki je predstavljena na sliki 4.9. Za vsak dan se generirajo naslednji deli:

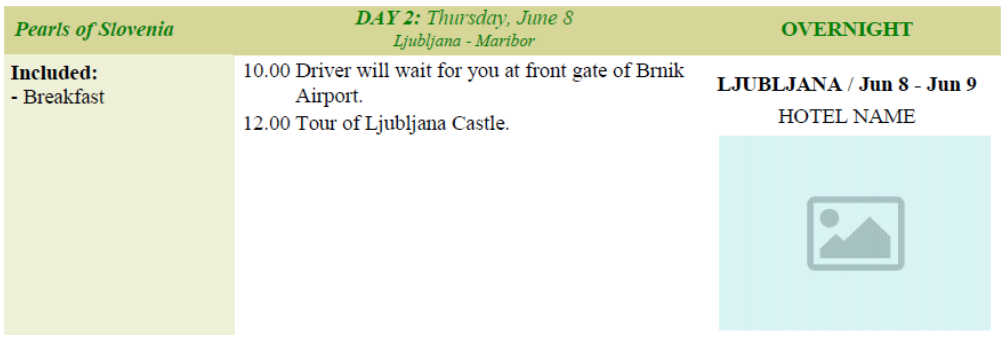
- desno zgoraj je naslov dneva;
- na sredini spodaj sta datum in pot;
- levo spodaj so naštete vključene storitve;
- na sredini spodaj so opisi, nasveti ter dodatne informacije;
- levo spodaj je mesto z datumi nočitev ter imenom hotela; pod njimi se izrišejo izbrane slike;



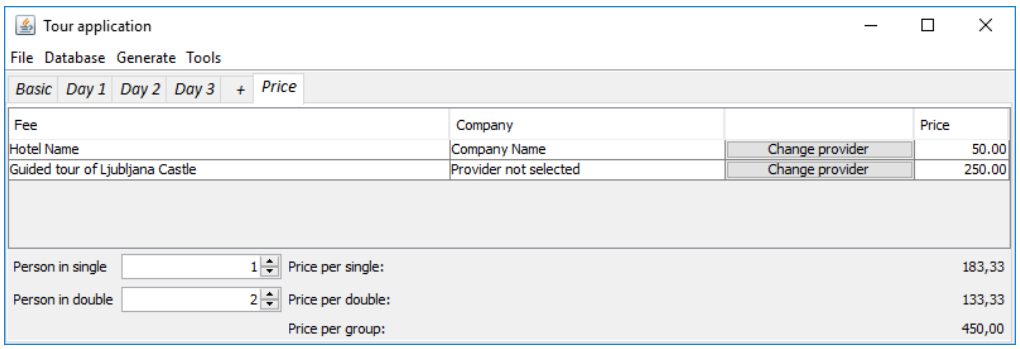
Slika 4.8 Pogled za vnos podatkov o dnevu. Na desni je polje v obliki tabele, ki skrbi za vnos opisa. Na levi so gumbi za izbiro poti, hotela, nasvetov ter slik.

Na koncu se izpišejo še dodatne informacije, kot so cene, pogoji uporabe ter daljši opisi izbranih hotelov. Vsi podatki so del predloge in so predhodno vpisani, ali pa se naložijo iz podatkovne baze.

Dialog za izvoz itinerarja v PDF format omogoča vnos cen in izbiro določenih dodatnih nastavitev izpisa, kot je izpis brez ur in pa izpis dodatnih informacij o hotelih, ki se naložijo iz podatkovne baze. Prednost PDF formata je, da ga je mogoče programsko izdelati in ima enak videz na vseh sistemih. PDF format ima namensko aplikacijo, ki omogoča dodatne funkcionalnosti, kot sta digitalno podpisovanje in tiskanje.



Slika 4.9 Pogled za vnos podatkov o dnevu. Na desni je polje v obliki tabele, ki skrbi za vnos opisa. Na levi so gumbi za izbiro poti, hotela, nasvetov ter slik.



Slika 4.10 Pogled servisov s pripadajočimi cenami v aplikaciji.

Personalizacijo itinerarja za naročnika aplikacije naredi razvijalec in je brez njegove pomoči ni mogoče naknadno spreminjati. To pripomore k temu, da sta implementacija ter zagotavljanje izgleda lažja, obenem pa je otežena nenadzorovana uporaba programa. Posredno uporabnika s tem ko mu onemogočimo majhne popravke v končnem izdelku prisilimo, da se drži zastavljene oblike.

4.2.5 Stroški itinerarja

Oceno stroškov je mogoče spremljati ob vnosu podatkov iz itinerarja v namenskem pogledu. Podatke o cenah program dobi iz podatkovne baze in jo računa na podlagi izbranih servisov za posamezen dan. Omogoča vnos števila oseb in s tem bolj natančno računanje stroškov. Pogled, ki prikazuje oceno stroškov, je predstavljen na sliki 4.10.

Za namene kasnejše uporabe in izvoza je mogoče urejati podatke o imenih in cenah ter za vsak servis izbrati podjetje, ki bo storitev izvedlo, oziroma se to prebere iz podatkovne baze.

Sistem omogoča izvoz podatkov o stroških s podatki iz tega pogleda v format Microsoft Excel. Ob izvozu se izvede tudi olepšava podatkov, saj program naredi optimizacije kot je združitev več nočitev v posameznem hotelu in prilagoditve formul v Microsoft Excel-u. Ta dokument je primeren za uporabo v kasnejših korakih priprave potovanja, saj je primeren za deljenje, obenem pa je format Microsoft Excel veliko bolj priročen za obdelavo podatkov.

4.2.6 Shranjevanje

Predlagana rešitev omogoča shranjevanje vnešenega itinerarja v datoteko. Za te datoteke se uporablja končnica *nsx*. Stanje programa se shrani v XML označevalnem jeziku in se

nato obnovi ob odprtju datoteke. Lastna končnica sistemu omogoča, da se lahko nastavi aplikacijo kot privzeto za odpiranje *nsx* datotek.

Shranjevanje sistemu doprinese ponovno poenostavitev procesa, saj se lahko s kopiranjem iz obstoječega potovanja naredi novo potovanje. Potrebna je zgolj sprememba osnovnih podatkov in itinerar je pripravljen za novo stranko. Obenem pa so s shranjevanjem mogoče iteracije potovanja, saj se vnašajo zgolj spremembe in prilagoditve željam stranke. Mogoče je voditi tudi več različic itinerarja.

4.2.7 Pakiranje aplikacije

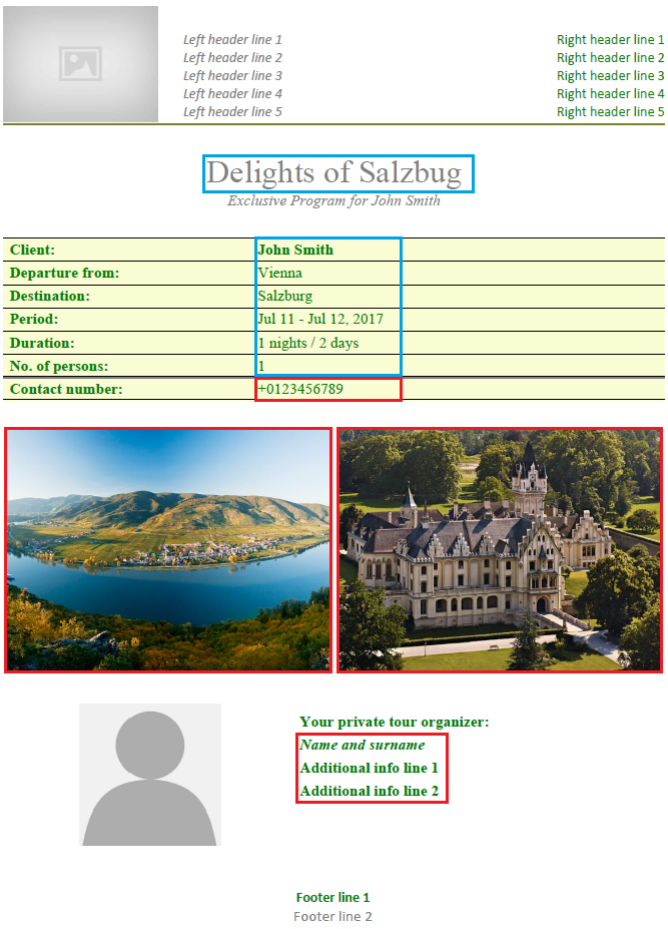
Programske datoteke aplikacije so združene v JAR datoteko. To je arhiv, ki združi javanske razrede s pripadajočimi artefakti v izvršilno datoteko, ki jo zna poganjati javansko izvajalno okolje. Obenem so za poganjanje aplikacije potrebne še dodatne knjižnice, ki jih z glavno JAR datoteko povezuje manifest datoteka. Manifest datoteka vsebuje vse informacije ter metapodatke o JAR datoteki in odvisnih knjižnicah.

Aplikacijo s potrebnimi knjižnicami je mogoče shraniti v mapo na Raspberry Pi-ju in s tem omogočiti njeno uporabo vsem uporabnikom na omrežju. Takšna raba omogoča uporabo brez nalaganja, saj je potrebno zgolj javansko izvajalno okolje in mogoč je zagon datoteke, ki se nahaja na mrežni lokaciji. Dodaten plus takšne uporabe je enostavno posodabljanje programa, saj je potrebno zamenjati zgolj datoteko na mrežni lokaciji.

4.3 Avtomatizacija sestave itinerarja

Generacija PDF dokumenta z itinerarjem, ki predstavlja glavni produkt procesa sestave, je skoraj popolnoma avtomatizirana. Izgled prve strani je predstavljen na sliki 4.11. Generacija delov označenih z rdečo barvo zahteva zgolj izbiro primerne entitete v bazi, s pomočjo katere se nato generira itinerar s podatki iz podatkovne baze. Generacija delov označenih z modro barvo zahteva ročni vnos ustreznih podatkov v aplikacijo. To so predvsem podatki, ki so specifični za vsako potovanje. Neoznačeni deli itinerarja so del predloge in se ne spreminjajo. Iz slike je razvidno, da je potrebno vnesti zgolj peščico podatkov.

Izgled opisa dni, ki predstavlja glavni del itinerarja, je predstavljen na sliki 4.12. Ponovno je razvidno, da je večina podatkov pridobljenih iz podatkovne baze. Predstavljene slike nam nazorno orišejo, katere podatke je treba vnesti ročno. Ti podatki vključujejo predvsem:



Slika 4.11 Izgled prve strani itinerarja v praksi.

- podatke o stranki,
- lokacijo odhoda,
- destinacijo,
- čas trajanja,
- podatek o številu oseb,
- naslove dni itd.

Podatki, ki jih pridobimo iz podatkovne baze glede na izbrane entitete, so sledeči:

- slike,

Slika 4.12 Izgled druge strani itinerarja, ki vsebuje opise dni v praksi.

- kratke opise dejavnosti,
- daljše opise dejavnosti na določen dan,
- podatke o hotelu,
- nasvete itd.

Večino avtomatsko pridobljenih podatkov je mogoče ročno popravljati, vendar aplikacija teži k temu, da je takšnih popravkov čimmanj. Ročni popravki namreč občutno podaljšajo čas izdelave itinerarja.

4.4 Analiza težav sistema v praksi

Opisana rešitev je cenovno ugodna in primerna za manjše število uporabnikov. Neupoštevanje teh omejitev lahko privede do različnih težav. Na srečo je večino težav mogoče odpraviti s predhodnim planiranjem in oceno uporabe sistema. V primeru večjih obremenitev je namreč priporočljiva uporaba strežnika močnejšega od Raspberry Pi-ja in pa bolj zmogljivih pomnilnih naprav kot je SD kartica.

4.4.1 Izraba spominske kartice

Raspberry Pi je izredno uporaben, vendar pa je ena od njegovih glavnih prednosti tudi ena od njegovih glavnih slabosti. Za operacijski sistem in hrambo podatkov uporablja SD kartico. V primerjavi s trdimi diski, ki jih uporabljajo osebni računalniki, ima bistveno manjše predvideno število branj in pisanj preden pomnilna celica odpove. Število branj in pisanj, ki jih Raspberry Pi naredi, je sorazmerno z njegovo uporabo. To pomeni, da več kot sistem uporabljamo, hitreje SD kartica neha delovati. V primeru manj kvalitetnih kratic se to lahko zgodi že v parih mesecih. Zamenjava kartice sicer ne predstavlja velikega stroška, predstavlja pa izpad delovanja sistema in dodatno delo.

4.4.2 Prevelika obremenitev

Druga težava se pojavi v primeru intenzivne uporabe Raspberry Pi-ja, saj veliko število sočasnih uporabnikov upočasni sistem in posledično delovanje aplikacije, kar privede do neprijetne uporabniške izkušnje ali celo nedelovanja sistema.

4.4.3 Večje količine podatkov

Količina podatkov na Raspberry Pi-ju je omejena z velikostjo SD kartice. To pomeni, da ima v praksi na voljo par gigabajtov podatkov. Zaradi predhodno naštetih težav nakup velike SD kartice ni priporočljiv, saj se sistem hitro preobremeni in SD kartica izrabi.

4.4.4 Varnost podatkov

Največjo vrednost sistema predstavljajo prav podatki shranjeni na SD kartici bodisi v obliki datotek ali v podatkovni bazi. Zaradi omenjenih težav in drugih nepredvidljivih dogodkov je pomembno, da se dovolj pogosto ustvarjajo varnostne kopije podatkov. Idealno je dnevno kopiranje podatkov na zunanji medij, ki je v primeru Raspberry Pi-ja

lahko USB ključ ali zunanji disk. To se naredi v obliki ponavljajoče naloge na operacijskem sistemu, ki zažene skripto za kopiranje vseh podatkov. Za nemoteno delo s sistemom se ta naloga izvaja izven delovnega časa.

4.5 Problem pomnjenja

Raspberry Pi v okviru trenutne rešitve kot glavni pomnilni medij uporablja SD kartico Kingston Ultimate 233x. Spominska kartica ima naslednje lastnostni [18]:

- 16 GB prostora,
- 90 MB/s hitrost branja,
- 45 MB/s hitrost pisanja,
- 1,4 ms zakasnitev bralnega dostopa,
- 4,2 ms zakasnitev pisalnega dostopa.

Zgoraj naštete vrednosti branja in pisanja so nominalne kar pomeni, da jih je mogoče doseči zgolj v idealnih pogojih. V praksi se izkaže da hitrost branja doseže do 45 MB/s, hitrosti pisanja pa do 35 MB/s.

4.5.1 Vrste SD pomnilnih kartic

Glavni problem SD kartic je predvideno število bralno-pisalnih ciklov, ki pri SD karticah višje kakovosti doseže do 100 000 ciklov na celico, pri slabših pa le desetino te vrednosti. Število bralno pisalnih ciklov določa tip pomnilne celice uporabljene pri izdelavi SD kartice. SD kartice danes uporabljajo tri tipe pomnilnih celic in sicer SLC, MLC in TLC. SLC pomnilne celice imajo dve stanji in pomnijo 1 bit podatkov, MLC pomnilne celice imajo 4 stanja in pomnijo 2 bita podatkov, TLC pomnilne celica pa imajo 8 stanj in hranijo kar 3 bite podatkov. Več bitov kot SD kartica hrani v eni celici, manj bralno-pisalnih ciklov dočaka. SD kartice namenjene vsakodnevni rabi so ponavadi narejene s TLC pomnilnimi celicami, ki so najcenejše. Temu primerna je tudi njihova življenjska doba.

4.5.2 Možne nadgradnje pomnilnega segmenta

V prejšnjih razdelkih je bilo omenjeno, da imajo SD kartice v primerjavi z magnetnimi diski krajšo življenjsko dobo. Ta problem je mogoče premostiti tako, da na SD kartici teče samo operacijski sistem, vse ostale storitve pa uporabljajo zunanje pomnilne medije.

USB zunanji trdi disk

Ena od možnih rešitev je uporaba zunanjega trdega diska, ki ga na Raspberry Pi povežemo preko USB vhoda. Težave pri uporabi zunanjega diska prek USB vhoda nastanejo v primeru, ko zunanji trdi disk nima lastnega napajanja ampak se napaja prek USB vhoda. V takšnem primeru se lahko pojavi težava, da napajanje Raspberry Pi-ja ne zadostuje potrebam sistema, kar povzroči nedelovanje sistema. V primeru zunanjega trdega diska predstavlja problem tudi ustvarjanje varnostnih kopij, saj ponavadi te funkcionalnosti ne omogočajo.

Mrežni trdi disk

Problem je mogoče rešiti tudi s trdim diskom priklopljenim v omrežje. Prednost te rešitve je, da takšni sistemi omogočajo RAID[2] diskovna polja. RAID omogoča varnostno kopiranje podatkov z uporabo več diskov, obenem pa omogoča višje hitrosti branja in pisanja. Pri uporabi takšne rešitve hitrost delovanja omejuje hitrost prenosa preko omrežja.

[2]RAID - redundant array of independent disks

5 Zaključek

V pričujočem delu smo v obliki izdelane arhitekture odjemalec-strežnik avtomatizirali proces sestave turističnega itinerarja. Končno rešitev sestavljata strežnik in aplikacija, ki teče na odjemalcih. Za strežnik smo uporabili Raspberry Pi, odjemalca pa predstavlja vsak računalniški sistem povezan v isto omrežje kot strežnik, ki podpira Javo 8. Z izpolnjevanjem slednjega pogoja lahko računalniški sistem poganja aplikacijo odjemalca, ki je napisana v Javi. Glavne prednosti implementirane rešitve so:

- cenovna dostopnost,
- nizke strojne zahteve,
- nizki stroški vzdrževanja,
- nadzor nad lastnimi podatki,
- enostavna uporaba,
- hitro delovanje,
- prilagodljivost obstoječim procesom.

Uporaba rešitve sprva ne prinese vidnih pohitritev dela, saj je potrebno podatkovno bazo napolniti s podatki in iz izdelanih itinerarjev shraniti predloge dni. Z dodatnim delom je sprva izdelava itinerarja lahko celo počasnejša, kot bi bila brez uporabe rešitve. Dolgoročno se dodatno delo obrestuje, ker je mogoče s pravilno uporabo celoten postopek drastično skrajšati. Podatkovna baza, napolnjena s podatki, omogoča hitro sestavo itinerarjev in izračun stroškov. Obenem skuša uporabniku pomagati na ta način, da enkrat vnešene podatke večkrat uporabi.

Izbrane tehnologije so nam omogočile uspešno implementacijo in vzpostavitev sistema. Strežniški sistem Raspberry Pi se je v praksi izkazal za dobro izbiro, ob upoštevanju omejitev njegove uporabe. Uporabniška aplikacija tudi v praksi pospeši proces izdelave itinerarja in se dobro prilagaja procesom znotraj manjše turistične agencije. Izkazalo se je, da največji problem za uspešno uporabo aplikacije predstavlja učenje uporabnikov, kako pravilno uporabljati njene funkcionalnosti in si tako čim bolj olajšati delo.

Delujoča rešitev se je izkazala kot uspešna. V nadaljevanju bi bilo v sistem potrebno vključiti še nadaljnje korake organizacije potovanja. To vključuje izdelavo druge dokumentacije za stranko in organizacijo potovanja iz operativnega vidika.

Spletne predstavitve so vedno bolj zanimive in fleksibilne v primerjavi s klasičnimi dokumenti. Možnost izdelave končnega itinerarja v spletni obliki bi predstavljala dodano vrednost. Spletna predstavitev bi omogočila integracijo s storitvami kot so zemljevidi, olajšala dostopnost do itinerarjev za stranke in možnost, da stranka vidi popravke takoj, ko so vnešeni.

Izboljšava bolj raziskovalne narave pa bi bila popolna avtomatizacija sestave itinerarja. S pomočjo strojnega učenja bi bilo mogoče izdelavo popolnoma avtomatizirati. Sistem bi se lahko učil na podlagi ročno izdelanih itinerarjev in podatkov v podatkovni bazi. Sestava itinerarja je namreč iskanje kombinacij po zahtevah stranke.

Literatura

[1] iTravel - travel agency software, Dosegljivo: http://www.itravelsoftware.com, [Dostopano: Marec 2017].

[2] Travefy - Wikipedia, Dosegljivo: https://en.wikipedia.org/wiki/Travefy, [Dostopano: Marec 2017].

[3] Aplikacije 'odjemalec - strežnik', Dosegljivo: http://colos1.fri.uni-lj.si/ERI/RACUNALNISTVO/PODATKOVNE_BAZE/aplikacije_odjemalec__strenik.html, [Dostopano: April 2017].

[4] Raspberry Pi - Wikipedia, Dosegljivo: https://en.wikipedia.org/wiki/Raspberry_Pi, [Dostopano: April 2017].

[5] Ten millionth Raspberry Pi, and a new kit - Raspberry Pi, Dosegljivo: https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/, [Dostopano: Marec 2017].

[6] W. L. Rosch, Hardware Bible Fifth Edition, Macmillan Computer Publishing (a Pearson Education company), Indianapolis, 1999.

[7] Raspberry Pi 3 Model B - Raspberry Pi, Dosegljivo: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, [Dostopano: April 2017].

[8] Raspberry Pi FAQs - Frequently Asked Questions, Dosegljivo: https://www.raspberrypi.org/help/faqs/, [Dostopano: April 2017].

[9] RaspbianAbout - Raspbian, Dosegljivo: https://www.raspbian.org/RaspbianAbout, [Dostopano: April 2017].

[10] When to use NoSQL vs SQL, Dosegljivo: https://docs.microsoft.com/en-us/azure/documentdb/documentdb-nosql-vs-sql, [Dostopano: April 2017].

[11] SQL - Wikipedia, Dosegljivo: https://en.wikipedia.org/wiki/SQL, [Dostopano: April 2017].

[12] MySQL :: Why MySQL?, Dosegljivo: https://www.mysql.com/why-mysql/, [Dostopano: April 2017].

[13] TIOBE Index — TIOBE - The Software Quality Company, Dosegljivo: https://www.tiobe.com/tiobe-index/, [Dostopano: Marec 2017].

[14] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, The Java® Language Specification - Java SE 8 Edition, Dosegljivo: http://docs.oracle.com/javase/specs/jls/se8/html/index.html, [Dostopano: Marec 2017].

[15] What are the system requirements for Java?, Dosegljivo: https://www.java.com/en/download/help/sysreq.xml, [Dostopano: April 2017].

[16] Java™ Platform, Standard Edition 8 API Specification, Dosegljivo: https://docs.oracle.com/javase/8/docs/api/, [Dostopano: Marec 2017].

[17] Hibernate ORM, Dosegljivo: http://hibernate.org/orm/, [Dostopano: April 2017].

[18] Benchmark Results: Access Time And I/O Performance, Dosegljivo: http://www.tomshardware.com/reviews/sdxc-sdhc-uhs-i,2940-9.html, [Dostopano: Junij 2017].